CLAIMS

1. A cryptographic calculation method characterised in that, in order to execute a block of instructions (Π_j) chosen as a function of an input variable (D_i) from amongst N predefined blocks of instructions (Π_1, …, Π_N), a block ($\Gamma(k,s)$) common to the N predefined blocks of instructions (Π_1, …, Π_N) is executed a predefined number (L_j) of times, and the predefined number (L_j) being associated with the chosen block of instructions (Π_j).

2. A method according to Claim 1, in which the predefined number (L_j) is variable from one predefined block of instructions (Π_1, …, Π_N) to another.

3. A method according to one of Claims 1 to 2, in which the common block ($\Gamma(k,s)$) comprises at least one calculation instruction (γk) equivalent, vis-à-vis a covert channel attack, to a calculation instruction of each predefined block (Π_1, …, Π_N).

4. A method according to Claim 3, in which the common block ($\Gamma(k,s)$) also comprises an instruction to update a loop pointer (k) indicating a number of executions already executed of the common elementary block ($\Gamma(k,s)$).

5. A method according to Claim 3 or Claim 4, in which the common block ($\Gamma(k,s)$) also comprises an instruction to update a state pointer (s) indicating whether the predefined number (L_j0 has been reached.

6. A method according to Claim 4 or Claim 5, in

which the value of the loop pointer (k) and/or the value of the state pointer (s) are a function of the value of the input variable (D_i) and/or of the number of instructions of the block of instructions (P_j) associated with the input data value.

7. A method according to one of Claims 1 to 6, in which, in order to successively effect several blocks of instructions chosen from amongst the N predefined blocks of instructions (Π_1, ..., Π_N), each chosen block of instructions (Π_j) being selected as a function of an input variable (D_i) associated with an input index (i),

the common elementary block ($\Gamma(k,s)$) is executed a total number (L_T) of times, the total number (L_T) being equal to a sum of the predefined numbers (L_j) associated with each chosen block of instructions (Π_j).

8. A method according to Claim 7, during which one and the same block of instructions may be chosen several times according to the input variable associated with the input index (i).

9. A method according to one of Claims 7 or 8, in which the value of the loop pointer (k) and/or the value of the state pointer (s) and/or the value of the input variable (D_i) and/or the number of instructions of the block of instructions (Π_j) associated with the value of the input data item (D_i) are linked by one or more mathematical functions.

10. A method according to Claim 9, used in the implementation of an exponentiation calculation of the

type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following steps:

```
Initialisation:
        $R_0$ <- 1; $R_1$ <- A; i <- M-1
As long as i ≥ 0, repeat the common block $\Gamma(k,s)$:
        k <- (/s)x(k+1) + sx2x(/$D_i$)
        s <- (k mod 2) + (k div 2)
        $\gamma(k,s)$:      $R_0$ <- $R_0 x R_{k \bmod 2}$
                            i <- i - s
Return $R_0$.
```

11. A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following steps:

```
Initialisation:
        $R_0$ <- 1; $R_1$ <- A; i <- M-1; k <- 1
As long as i ≥ 0, repeat the common block $\Gamma'(k,s)$:
        k <- ($D_i$) AND (/k)
        $\gamma'(s,k)$:     $R_0$ <- $R_0 x R_k$
                            i <- i - (/k)
Return $R_0$.
```

12. A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input

index i, the method comprising the following steps:

```
Initialisation:
    $R_0$ <- 1; $R_1$ <- A; i <- 0; k <- 1
As long as $i \leq M-1$, repeat the block $\Gamma(k,s)$:
    k <- k $\oplus$ $D_i$
    $\gamma(k)$: $R_k$ <- $R_k x R_1$
          i <- i+k
Return $R_0$.
```

13. A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following steps:

```
Initialisation:
    $R_0$ <- 1; $R_1$ <- A,; $R_2$ <- $A^3$;
    $D_{-1}$ <- 0; i <- M-1; s <- 1
As long as $i \geq 0$, repeat the block $\Gamma(k,s)$:
    k <- (/s)x(k+1) + sx($D_i$ + 2x($D_i$ AND $D_{i-1}$))
    s <- /((k mod 2) $\oplus$ (k div 4))
    $\gamma(k,s)$:    $R_0$ <- $R_0 x R_{sx(k \text{ div } 2)}$
                i <- i - sx(k mod 2 + 1)
Return $R_0$.
```

14. A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following steps:

```
Initialisation:
    $R_0$ <- 1; $R_1$ <- A; $R_2$ <- $A^3$;
    $D_{-1}$ <- 0; i <- M-1; s <- 1
As long as i ≥ 0, repeat:
    k <- (/s)x(k+1)
    s <- s $\oplus$ $D_i$ $\oplus$ (($D_{i-1}$ AND (k mod 2))
    $\Gamma$(k,s):    $R_0$ <- $R_0$x$R_{kxs}$
                      i <- i - kxs - (/$D_i$)
Return $R_0$.
```

15. A method according to one of Claims 7 or 8, in which the links between the value of the loop pointer (k) and/or the value of the state pointer (s) and/or the value of the input variable (D_i) and/or the number of instructions of the block of instructions (Π_j) associated with the value of the input data item (D_i) are defined by a table with several inputs such as a matrix (U(k,1)).

16. A method according to Claim 15, used in the implementation of an exponentiation calculation of the type $B = A^D$, D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following step:

```
As long as i ≥ 0, repeat the block $\Gamma$(k,s):
    k <- (/s)x(k+1) + sx2x(/$D_i$)
    s <- U(k,1)
    $\gamma$(k,s):    $R_0$ <- $R_0$x$R_{U(k,0)}$
                      i <- i - s
```

where (U(k,1)) is the following matrix:

$$(U(k,1))_{\substack{0 \le k \le 2 \\ 0 \le 1 \le 1}} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

17. A method according to Claim 15, used in the implementation of an exponentiation calculation of the type $B = A^D$ according to the algorithm (M, M^3), D being an integer number of M bits, each bit (D_i) of D corresponding to an input variable of input index i, the method comprising the following step:

```
As long as $i \ge 0$, repeat the common block $\Gamma(k,s)$:
        $k \leftarrow (/s)x(k+1) + sx(D_i + 2x(/D_i \text{ AND } D_{i-1}))$
        $s \leftarrow U(k,2)$
        $\gamma(k,s)$:      $R_0 \leftarrow R_0xR_{U(k,0)}$;
                            $i \leftarrow i - U(k,1)$
```

where (U(k,1)) is the following matrix:

$$(U(k,1))_{\substack{0 \le k \le 5 \\ 0 \le 1 \le 2}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 2 & 1 \end{pmatrix}.$$

18. A method according to Claim 15, used in the implementation of a calculation on an elliptic curve in affine coordinates, a calculation using operations of the addition or doubling of points type, and in which the following step is performed:

```
As long as $i \geq 0$, repeat $\Gamma(k,s)$:
        $\gamma(k)$: $R_{U(k,0)}$ <- $R_1 + R_3$;
                $R_{U(k,1)}$ <- $R_{U(k,1)} + R_{U(k,2)}$;
                $R_5$ <- $R_2/R_1$; $R_{U(k,3)}$ <- $R_1 + R_5$;
                $R_{U(k,4)}$ <- $R_5^2$;
                $R_{U(k,4)}$ <- $R_{U(k,4)} + a$;
                $R_1$ <- $R_1 + R_{U(k,5)}$;
                $R_2$ <- $R_1 + R_{U(k,6)}$; $R_6$ <- $R_1 + R_{U(k,7)}$;
                $R_5$ <- $R_5 . R_6$; $R_2$ <- $R_2 + R_5$
        s <- $k - D_i + 1$
        k <- (k+1) x (/s);
        i <- i - s;
```

where (U(k,1)) is the following matrix:

$$(U(k,1))\ \begin{matrix} 0 \leq k \leq 1 \\ 0 \leq 1 \leq 10 \end{matrix} = \begin{pmatrix} 1 & 2 & 4 & 1 & 6 & 6 & 4 & 3 \\ 6 & 6 & 3 & 5 & 1 & 5 & 2 & 6 \end{pmatrix}.$$

19. A method for obtaining an elementary block ($\Gamma(k,s)$) common to N predefined blocks of instructions (Π_1, …, Π_N), a method able to be used for implementing a cryptographic calculation method according to one of Claims 1 to 12, the method being characterised in that it comprises the following steps:

E1: breaking down each predefined block of instructions (Π_1, …, Π_N) into a series of elementary blocks (γ) equivalent vis-à-vis a covert channel attack, and classifying all the elementary blocks,

E2: seeking a common elementary block ($\gamma(k,s)$) equivalent to all the elementary blocks (γ) of all the predefined blocks of instructions,

E3: seeking a common block ($\Gamma(k,s)$) comprising at least the common elementary block ($\gamma(k,s)$) previously obtained and an instruction to update a loop pointer (k) such that an execution of the common elementary block associated with the value of the loop pointer (k) and an execution of the elementary block with a rank equal to the value of the loop pointer (k) are identical.

20. A method according to Claim 19, characterised in that, during step E1, at least one fictional instruction is added to at least one predefined block of instructions.